

Controller re-specification to minimize switching activity in controller/data path circuits

Anand Raghunathan [†], Sujit Dey [‡], Niraj K. Jha [†], and Kazutoshi Wakabayashi [§]

[†]Department of EE, Princeton University, Princeton, NJ 08544

[‡]C&C Research Labs, NEC USA, Princeton, NJ 08540

[§]C&C Research Labs, NEC Corp., Tokyo, Japan

ABSTRACT: This paper proposes a controller-based technique for minimizing switching activity in controller/data path circuits. Though the control signals in a register transfer level (RTL) implementation are fully specified, they can be re-specified under certain states/conditions when the data path components that they control need not be active. Unlike techniques that insert extra circuitry like transparent latches, controller re-specification is a low-overhead technique that merely reconfigures existing multiplexer networks and functional units to minimize activity in the data path. Hence, it is well suited to control-flow intensive designs, where power consumption in multiplexer networks forms a major component of the total power consumption. Our controller re-specification algorithm consists of constructing an activity graph for each data path component, identifying conditions under which the component need not be active, and re-labeling the activity graph resulting in re-specification of the corresponding control expressions. Application of the proposed technique to several RTL circuits demonstrated the ability to reduce the total (controller + data path) power consumption by up to 51.8% compared to the initial area-optimized implementations, with nominal area and delay overheads.

I. Introduction

Power consumption in CMOS circuits is dominated by the dynamic component that is incurred whenever signals in the circuit undergo logic transitions. In practice, a large portion of the signal transitions that occur in a circuit are unnecessary, *i.e.*, they have no effect on the value at the circuit output. Recognizing this fact, several techniques have been proposed to reduce power consumption by eliminating unnecessary transitions at various signals in the circuit. The idea of shutting down unused parts of a circuit has been used by designers for a long time, and is employed in most modern microprocessors and microcontrollers that target portable applications [1, 2]. Various automated synthesis techniques exploiting this idea in several ways have also been investigated [3, 4, 5, 6, 7].

This paper presents a controller re-specification technique to reduce power consumption in controller/data path circuits. This technique re-designs the logic that generates the control signals to the data path such that activity in various data path components including internal signals of multiplexer trees and inputs to functional units and comparators is reduced. Like several other previously proposed techniques that insert extra circuitry (*e.g.* transparent latches) to reduce activity, our technique also exploits idle conditions for various data path components. However, the extra circuitry inserted can itself incur significant area and power costs which is well justified only when large blocks of logic, like functional units, are shut off. On the other hand, controller re-specification is a low-overhead technique that reconfigures the existing multiplexer networks and functional

units in the data path (by re-designing their control logic) so that unnecessary activity is minimized. Moreover, while most known techniques seek to completely eliminate activity in the idle logic, our method may significantly reduce but not completely eliminate such activity. As a result, it is frequently possible to reduce activity while avoiding the overheads associated with conventional shut-off based techniques. We have found controller re-specification to be well suited to control-flow-intensive designs, where the multiplexer power consumption constitutes a major part of the total power consumption [8].

II. Controller re-specification to reduce power

We illustrate the effect of re-specifying control signals on the activity of data path signals through the following example. Consider an RTL circuit that implements the send process of the X.25 protocol [9] that is shown in Figure 1. Figure 2(a) shows an extracted part of the X.25 data path that consists of an ALU and the multiplexer trees that feed it. Figure 2(b) shows (i) the logic expressions for the control signals that feed the ALU and its multiplexer trees (xi represents a decoded state variable, *i.e.*, $xi = 1$ when the controller is in state Si), and (ii) an *activity graph* for the ALU, that indicates the operations performed by the ALU in each controller state. The vertices and arcs in the activity graph correspond to the controller states and state transitions. Each vertex in the activity graph is labeled with the computation performed by the ALU in the corresponding controller state. For example, consider controller state $S2$. Control signals $Sel(0)$, $Sel(1)$, $Sel(2)$ and $Selectfunc(0)$ (function select input, $1 \Rightarrow +$, $0 \Rightarrow -$) assume the logic values 1, 0, 1, and 0, respectively. From these values, it can be easily seen that the ALU performs the operation $bytes - byteCount$ in state $S2$.

In some states, like $S0$, the ALU may not be required to perform any operation, *i.e.*, its result may be unused. Using the scheduling and assignment information from high-level synthesis, it is possible to easily identify such idle states based on the absence of operations assigned to the ALU. In Figure 2(b), idle states are indicated by shaded vertices. The computation performed by the ALU in idle states can be changed without affecting the functionality of the design. The control signals $Sel(0)$, $Sel(1)$, $Sel(2)$, and $Selectfunc(0)$ were re-specified using the techniques that we present in the later sections. The resulting logic expressions for the re-specified control signals and the modified activity graph for the ALU are shown in Figure 2(c). Note that the labels of the vertices in the activity graph have changed. Consider two consecutive cycles in the operation of the X.25 circuit, during which the controller makes the state transition $S6 \rightarrow S4$. Under the original control expressions, there is switching activity in the ALU since its operands change from i (left operand), $byteCount$ (right operand), and 1 (function select input) in $S6$ to $Count$, One , and 0 respectively in $S4$. Under the re-specified control expressions, however, all input operands to the ALU remain stable. Hence, we conclude that re-specification of the control signals feeding the ALU, or equivalently, re-labeling of the ALU's activity graph, can affect the switching activity, and hence power consumption, in the ALU. During the actual operation of the circuit, controller state transitions other than $S6 \rightarrow S4$ will also occur. In general, it is necessary to consider all incoming and outgoing arcs while deciding on how to

This work was supported in part by NEC C & C Research Labs and in part by NSF under Grant No. MIP-9319269.

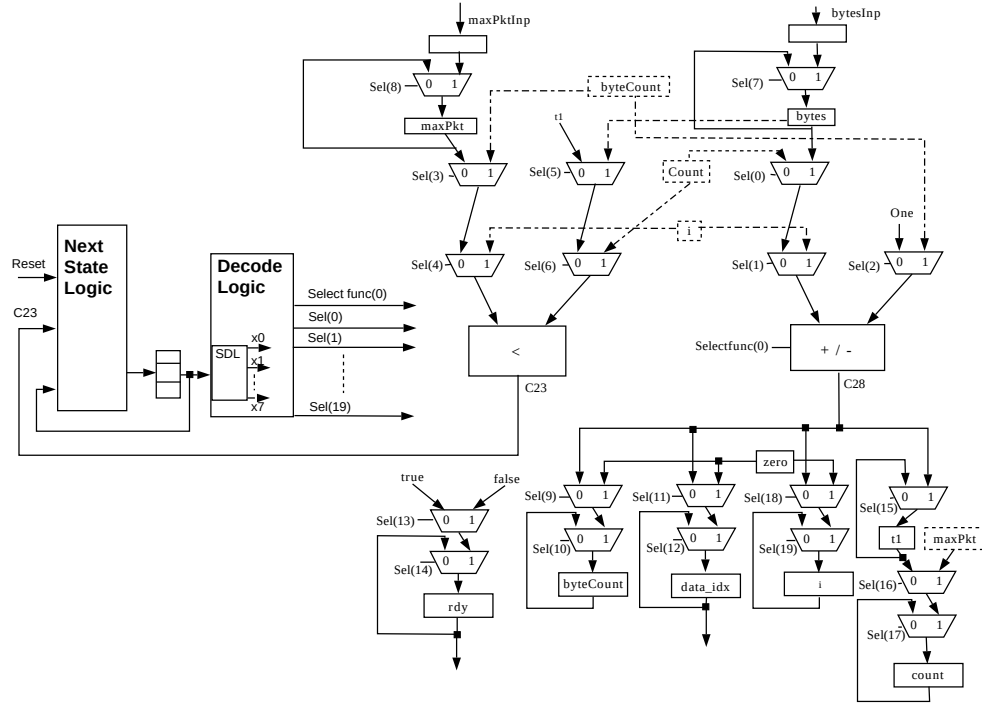


Figure 1: RTL circuit implementing the X.25 example

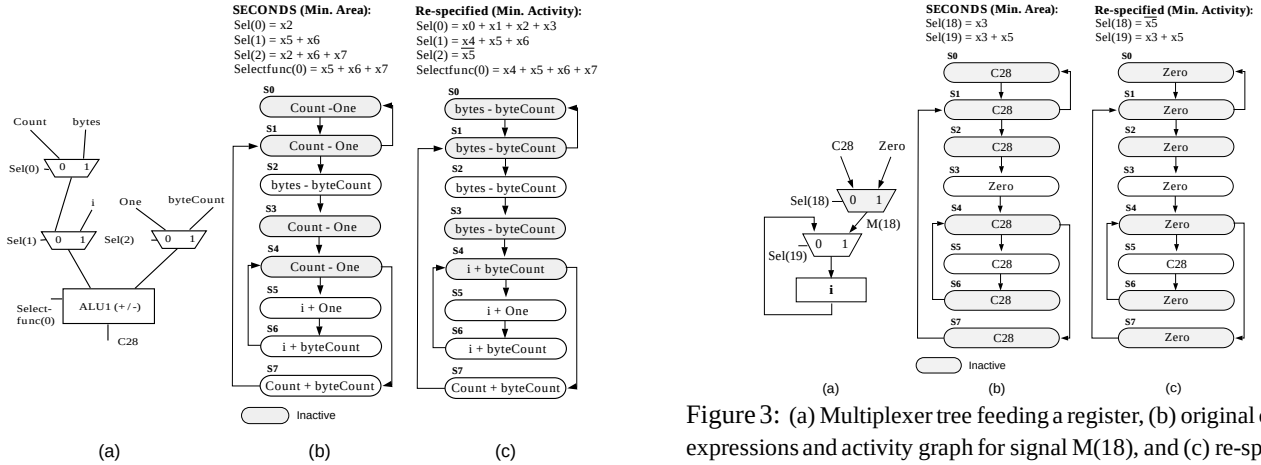


Figure 2: Control re-specification example (a) ALU and its multiplexer tree, (b) original control expressions and activity graph, and (c) re-specified control expressions and activity graph

(re-)label an idle vertex in the activity graph. We formalize the use of state transition probabilities or transition counts in labeling the activity graph later on in this section.

Re-specifying the controller can also lead to reduced activity within multiplexer trees. Figure 3(a) shows another part of the X.25 data path that consists of the register that stores variable i together with the multiplexers feeding it. The original expressions for the control signals $Sel(18)$ and $Sel(19)$ are given in Figure 3(b). The activity graph for signal $M(18)$ (the output of the shaded 2-to-1 multiplexer), that indicates which operand ($C28$ or $Zero$) is selected at $M(18)$, is also shown in Figure 3(b). The shaded vertices in the activity graph correspond to states when the value of signal $M(18)$ is not used. Consider controller state transition $S7 \rightarrow S1$. Since the computation performed by the ALU changes from $count + byteCount$ in $S7$ to $bytes - byteCount$ in $S1$ (see Figure 2(c)), the value of the operand ($C28$) itself changes. Re-specifying the

Figure 3: (a) Multiplexer tree feeding a register, (b) original control expressions and activity graph for signal $M(18)$, and (c) re-specified control expressions and activity graph

control signals as shown in Figure 3(c) eliminates this unnecessary activity at signal $M(18)$.

An alternative method for reducing activity in the multiplexer tree shown in Figure 3(a) is to insert transparent latches at the shaded multiplexer's data input corresponding to signal $C28$ and its select input, and "freeze" those transparent latches in states $S0$, $S1$, $S2$, $S4$, $S6$, and $S7$. The overhead incurred would be 9 transparent latches (note that $C28$ is an 8-bit signal), which constitutes a significant cost in area as well as power consumption relative to the modest power savings in the multiplexer tree. Indeed, when we inserted transparent latches as described above, and evaluated the circuit for power consumption after mapping it to a 1.0μ standard cell library (low power latches were used in order to minimize the power overhead), the total circuit power consumption decreased from $4.495mW$ initially to $4.461mW$ after inserting the shut-off circuitry, which corresponds to a very small decrease in power consumption. On the other hand, when the control signals were re-specified as shown in Figure 3(c), the total power consumption was reduced from $4.495mW$ to $4.291mW$. Generalizing to the

case of a b -bit m -to-1 multiplexer tree, it may be necessary to use $b * m + \log m$ transparent latches. To a first degree of approximation, we can see that the cost in terms of transparent latches as well as the potential for savings in terms of the amount of logic that can be shut-off scale similarly, indicating that using transparent latches is not likely to lead to significant reductions in power for larger multiplexers either. We verified the above hypothesis in practice for several multiplexer trees from our example RTL circuits.

A prominent characteristic of control-flow-intensive specifications is the presence of multiple mutually exclusive sequences (paths) of operations that may be performed in a single state. In order to fully exploit the opportunities for control signal re-specification, it is necessary to create a vertex in the activity graph corresponding to each mutually exclusive path in a state (also called a sub-state). However, in the presence of sub-states, re-specifying control signals can lead to a significant increase in the circuit delay, or may impact the glitching activity at control signals. These problems can be avoided by judicious re-specification of control signals [10].

Re-labeling the activity graph using state transition counts and activity matrices. Consider a less-than ($<$) comparator that is part of the data path in an RTL circuit and its partial activity graph shown in Figure 4. Vertex $S3$ in the activity graph corresponds to an idle state for the comparator. The activity graph also shows all vertices that have incoming arcs from or outgoing arcs to vertex $S3$. Vertices $S1$, $S2$, $S4$, and $S5$ have the labels $L1$ ($a < b$), $L2$ ($c < d$), $L2$, and $L3$ ($e < d$), respectively. We wish to label vertex $S3$ with one of the labels from the set $\{L1, L2, L3\}$ such that the activity, and hence the power consumption, in the comparator is minimized. For our illustrations in this paper, we use the expected (or average) bit transitions at the inputs of the comparator as a measure of its power consumption. However, it is possible to incorporate more sophisticated models for data path power consumption [11] to drive our controller re-specification technique.

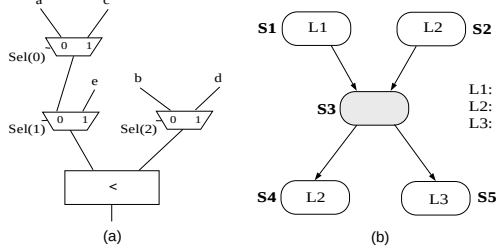


Figure 4: (a) Comparator and its multiplexer tree, and (b) activity graph used for re-labeling

For each arc $S_i \rightarrow S_j$ in the activity graph, $P(S_i \rightarrow S_j)$ is the probability of a controller state transition from S_i to S_j . An *activity matrix* $AM_{S_i \rightarrow S_j}$ for arc $S_i \rightarrow S_j$ in the activity graph stores the costs in terms of average bit transitions incurred for various combinations of labels that can be assigned to vertices S_i and S_j . The rows (columns) of $AM_{S_i \rightarrow S_j}$ are indexed by the set of possible labels that can be assigned to S_i (S_j). For example, consider the arc from vertex $S1$ to vertex $S3$ in the activity graph shown in Figure 4. Since the label of $S1$ is fixed to $L1$, the activity matrix $AM_{S1 \rightarrow S3}$ has valid entries in only one row, that is indexed by $L1$. We can (re-)label the vertices in the activity graph that correspond to idle conditions while attempting to minimize the *Total Labeling Cost* that is given by:

$$\text{Total Labeling Cost} = \sum_{\text{all } S_i \rightarrow S_j} AM_{S_i \rightarrow S_j}[L(S_i), L(S_j)].P(S_i \rightarrow S_j)$$

The expression $L(S_i)$ refers to the label assigned to vertex S_i in the activity graph, and $P(S_i \rightarrow S_j)$ refers to the probability of occurrence of the state transition $S_i \rightarrow S_j$. For the example of

```

Procedure CONTROL_RESPECIFY(RTL circuit  $R$ )
{
  IDENTIFY_RTL_BLOCKS( $R$ );
   $DDG \leftarrow \text{CREATE\_DATA\_DEPENDENCE\_GRAPH}(R)$ ;
  for each RTL block  $B$  in  $DDG$  in forward topological order {
     $U \leftarrow$  RTL unit in  $B$ ;
    STABILIZE_SIGNALS( $B, U$ );
    for each MUX tree  $T$  feeding  $U$  {
      STABILIZE_SIGNALS( $B$ , root of  $T$ );
    }
  }
}

Procedure STABILIZE_SIGNALS(RTL block  $B$ , circuit node  $N$ )
{
   $AG \leftarrow \text{CREATE\_ACTIVITY\_GRAPH}(B, N)$ ;
   $Labellist \leftarrow \text{FIND\_ALL\_LABELS}(AG)$ ;
  while (unlabeled vertices exist in  $AG$ ) {
    Select unlabeled vertex  $V^*$  such that
       $Labeled\_Vertex\_Arcs(V^*)$  is maximum;
    Select a label  $L^* \in Labellist$  such that
       $Incr\_Labeling\_Cost(V^*, L^*)$  is minimum;
    Assign label  $L^*$  to vertex  $V^*$ ;
  }
  if  $N$  is a MUX node {
    STABILIZE_SIGNALS( $B$ , left child of  $N$ );
    STABILIZE_SIGNALS( $B$ , right child of  $N$ );
  }
}

```

Figure 5: Overview of the controller re-specification algorithm

Figure 4, the cost of labeling vertex $S3$ with a label L^* is given by:

$$\begin{aligned} & AM_{S1 \rightarrow S3}[L1, L^*].P(S1 \rightarrow S3) + \\ & AM_{S2 \rightarrow S3}[L2, L^*].P(S2 \rightarrow S3) + \\ & AM_{S3 \rightarrow S4}[L^*, L2].P(S3 \rightarrow S4) + \\ & AM_{S3 \rightarrow S5}[L^*, L3].P(S3 \rightarrow S5) \end{aligned}$$

The objective, therefore, is to choose $L^* \in \{L1, L2, L3\}$ such the labeling cost as given by the above equation is minimized.

III. Controller re-specification algorithm

We have formulated controller re-specification as a Minimum Cost Vertex Labeling Problem (MCVLP) on the activity graph. We have shown that MCVLP is $\mathcal{NP}$ -Hard [10]. We use an efficient heuristic to solve the MCVLP problem.

Our controller re-specification algorithm is described by the pseudo-code shown in Figure 5. We start with an RTL specification that consists of a controller and data path. The data path is first broken up into its constituent *RTL blocks* by grouping *RTL units* (functional units, registers, and comparators) with the multiplexer networks that feed them. Then, we create the data dependence graph, where vertices represent RTL blocks, and arcs represent data dependencies. We visit RTL blocks by performing a topological ordering of the data dependence graph. If there is a data dependency from RTL block B_i to B_j , it is desirable to re-specify control signals for block B_i before block B_j . This is because a re-specification of the control signals feeding block B_i will affect the values appearing at its outputs, and hence the entries in the activity matrices of the activity graphs for block B_j . At each RTL block, we first attempt to re-specify control signals to minimize activity in its RTL unit. This is done by calling procedure STABILIZE_SIGNALS, passing to it

as a parameter the RTL unit in the current RTL block. Then, we call `STABILIZE_SIGNALS` once for each multiplexer network in the RTL block, passing to it as a parameter the root of the appropriate multiplexer network. `STABILIZE_SIGNALS` recursively traverses the multiplexer network, attempting to minimize activity at the output of the current multiplexer at each step.

We next describe procedure `STABILIZE_SIGNALS`, that performs the actual control signal re-specification to minimize the activity in a given functional unit or at an internal signal in a multiplexer network. First, we create an activity graph for the RTL unit or output of the multiplexer under consideration. We then use a simple heuristic procedure to solve MCVLP on the activity graph. The procedure iterates over the unlabeled vertices of the activity graph, labeling the vertex picked at each step so that the incremental cost of the labeling is minimized. The order in which unlabeled vertices are visited during the procedure has a significant impact on the final labeling. We choose an unlabeled vertex V^* in the activity graph to visit next, such that the quantity $Labeled_Vertex_Arcs(V^*)$ given by the following equation is maximized.

$$Labeled_Vertex_Arcs(V^*) = \sum_{\{V_i \rightarrow V^* \mid V_i \text{ is labeled}\}} P(V_i \rightarrow V^*) + \sum_{\{V^* \rightarrow V_j \mid V_j \text{ is labeled}\}} P(V^* \rightarrow V_j)$$

The rationale behind the criterion is that knowledge of the labels assigned to neighboring vertices is essential to accurately computing the cost of assigning a label to the picked vertex. Having picked vertex V^* to label next, we follow a greedy approach in assigning a label to it. We pick label $L^* \in L$ such that the quantity $Incr_Labeling_Cost(V^*, L^*)$ that is given by the equation below is minimized.

$$Incr_Labeling_Cost(V^*, L^*) = \sum_{\{V_i \rightarrow V^* \mid V_i \text{ is labeled}\}} AM_{V_i \rightarrow V^*}[L(V_i), L^*].P(V_i \rightarrow V^*) + \sum_{\{V^* \rightarrow V_j \mid V_j \text{ is labeled}\}} AM_{V^* \rightarrow V_j}[L^*, L(V_j)].P(V^* \rightarrow V_j)$$

IV. Experimental Results

In this section, we present results of the application of the proposed controller re-specification technique to RTL circuits implementing typical control-flow-intensive specifications. `Barcode` is a barcode reader preprocessor [12], `Change_maker` is one of the processes of a vending machine controller from [13], `Graphics` is a line-drawing process that is part of a graphics controller [10], and `X.25` is a simplified version of the `send` process of the `X.25` protocol [9]. The example designs are characterized by the presence of nested loops and conditionals, and array accesses in the behavioral descriptions, apart from arithmetic and comparison operations. The initial RTL circuits were obtained by synthesizing VHDL behavioral descriptions using the `SECONDS` [9, 14] behavioral synthesis system. The RTL circuits were optimized using NEC's `VARCHSYN` [15] logic synthesis system, and mapped to their commercial 1.0μ standard cell library. These mapped netlists were used to measure area, delay, and power consumption for the original designs. The RTL circuits were then passed to our controller re-specification procedure that re-specified the control expressions with an aim of reducing activity in the data path. The optimized RTL circuits were subject to the same logic synthesis and technology mapping optimizations as the original RTL circuit, and then evaluated for area, delay, and power consumption.

The results of our experiments are reported in Table 1. The columns **Original** and **Optimized** refer to the circuits before and

Table 1: Experimental results

Ckt.	Original			Optimized			Pow. Red. (%)
	Pow. (mW)	Area	Del. (ns)	Pow. (mW)	Area	Del. (ns)	
Bar-code	3.1	2004	49.4	2.7	1998	46.7	13.3
Change-maker	7.9	3700	70.6	4.8	3712	73.2	39.5
Graphics	4.7	3331	67.9	2.6	3341	65.3	45.2
X.25	4.5	2099	44.4	2.2	2102	49.6	51.8

after controller re-specification. The sub-columns **Pow.**, **Area**, and **Del.** provide the power consumption (in milliwatts), area (cell grid count + wiring area estimate), and delay (in nanoseconds) of the circuits after mapping to the technology library used. An in-house simulation-based power estimation tool [16] was used to measure power consumption. The vectors used for simulation were obtained from the behavioral test bench containing typical stimuli that were derived based on a knowledge of the design's functionality and intended use. The column **Pow. Red.** in Table 1 reports the percentage reduction in power consumption. The results indicate that significant savings in power consumption (upto 51.8%) are obtained with nominal overheads in area and delay.

References

- [1] A. Correale Jr., "Overview of the power minimization techniques employed in the IBM PowerPC 4xx embedded processors," in *Proc. Int. Symp. Low Power Design*, pp. 75–80, Apr. 1995.
- [2] J. Schulz, "A 3.3V 0.6um BiCMOS superscalar microprocessor," in *Proc. Int. Solid-State Circuits Conf.*, pp. 202–203, Feb. 1994.
- [3] L. Benini, P. Siegel, and G. DeMicheli, "Saving power by synthesizing gated clocks for sequential circuits," *IEEE Design & Test of Computers*, pp. 32–41, Winter 1994.
- [4] C. Papachristou, M. Spining, and M. Nourani, "A multiple clocking scheme for low power RTL design," in *Proc. Int. Symp. Low Power Design*, pp. 27–32, Apr. 1995.
- [5] M. Aldina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low power," *IEEE Trans. VLSI Systems*, vol. 2, pp. 426–436, Dec. 1994.
- [6] V. Tiwari, S. Malik, and P. Ashar, "Guarded evaluation: pushing power management to logic synthesis/design," in *Proc. Int. Symp. Low Power Design*, pp. 221–226, Apr. 1995.
- [7] E. Musoll and J. Cortadella, "High-level synthesis techniques for reducing the activity of functional units," in *Proc. Int. Symp. Low Power Design*, pp. 99–104, Apr. 1995.
- [8] A. Raghunathan, S. Dey, and N. K. Jha, "Register-transfer-level power optimization techniques with emphasis on glitch analysis and optimization," in *Proc. Design Automation Conf.*, June 1996.
- [9] S. Bhattacharya, S. Dey, and F. Brglez, "Performance analysis and optimization of schedules for conditional and loop-intensive specifications," in *Proc. Design Automation Conf.*, pp. 491–496, June 1994.
- [10] S. Dey, A. Raghunathan, N. K. Jha, and K. Wakabayashi, "Controller re-specification to minimize switching activity in controller/data path circuits," Tech. Rep. 96-C007-4-5016-1, NEC C&C Research Labs, Princeton, NJ, Mar. 1996.
- [11] P. Landman and J. M. Rabaey, "Architectural power analysis: the dual bit type method," *IEEE Trans. VLSI Systems*, vol. 3, pp. 173–187, June 1995.
- [12] High-level synthesis benchmarks, CAD Benchmarking Laboratory, Research Triangle Park, NC. Benchmarks can be downloaded from <http://www.cbl.ncsu.edu>.
- [13] D. L. Perry, *VHDL*. New York, NY 10020: McGraw-Hill, 1991.
- [14] S. Bhattacharya, S. Dey, and F. Brglez, "Clock period optimization during resource sharing and assignment," in *Proc. Design Automation Conf.*, pp. 195–200, June 1994.
- [15] *VARCHSYN Version 2.0 Users Manual*. Advanced CAD Development Laboratory, NEC Corporation, Nov. 1993.
- [16] *CSIM Version 5 Users Manual*. Systems LSI Division, NEC Corp., 1993.